SSTables and LSM Tree

Suresh Podeti
3 min read · May 16, 2023

In the previous article we have seen how dictionary or hash map is used to index databases.

Database indexing | Hash Indexes
In my previous article I have explained how database storage and retrieval works with an example:
medium.com

Log compaction and Merging
As we have seen in the previous articles database stores data by writing to an append-only log file as shown in Fig 1…
medium.com

Quick recap

Data is written to an append-only log file, after reaching a certain file size limit data is written to a new log file. In the background **log compaction and merging** (mentioned above in another article) process throws away duplicate keys in the log, and keeps only the most recent update for each key. After the merging process is complete, read requests are switched to use the new merged segment instead of the old segments, and then the old segment files can simply be deleted. **Each segment** now has its **own in-memory hash table**, mapping keys to file offsets. In order to find the value for a key, we first check recent segment's hash map; if the key is not present we check the second-most-recent segment, and so on.

Hash index has following limitations:

1. The hash table must fit in memory
2. Range queries are not efficient

In this article we will look at indexing structure that doesn't have those limitations, and focus more on one of the solution using SSTables (Sorted string table) and LSM-trees (Log-structured merge tree).

Each log-structured storage segment is a sequence of key-value pairs. These pairs appear in the order that they were written, and values later in the log take precedence over values for the same key in the log. Apart from that, the **order of key-value pairs in the file does not matter**. So, we can **sort sequence of key-value pairs by key.**

Constructing and maintaining SSTables

Maintaining a sorted structure on disk is possible, but maintaining it in memory is much easier. There are plenty of well-known tree data structures we can us, such as

> ***red-black tree or AVL trees.*** *With these data structures, we can insert keys in any order and read them back in sorted order.*

Our storage engine works as follows:

1. When write comes in, add it to an in-memory balanced tree data structure such as **red-black tree**. This in-memory tree is sometimes called a ***memtable***.

2. When the tree gets bigger than some threshold, write it out to disk as an SSTable file.

3. In order to serve a read request, first try to find the key in the ***memtable,*** then in the most recent on-disk segment, then in the next-older segment, etc.

4. From time to time, run a **merging and compaction** process in the background to combine segment files and to discard overwritten or deleted values. Merging segments is more efficient due to the records being pre-sorted. It uses merge step (mentioned in another article below) of merge sort for merging operation

Merge step in Merge sort

Merge sort is a divide-and-conquer algorithm based on the idea of breaking down a list into several sub-lists until…

medium.com

With keys sorted, you **don't need to have every single key in the index** anymore. If the key `B` is known to be somewhere between keys `A` and `C` you could just do a scan. This also means that **range queries are possible.**

An **SSTable** is a **key-sorted append-only key-value storage**. An **LSM-tree** is a layered data structure, based on a **balanced tree**, that **allows SSTables to exist without the controversy of being both sorted and append-only at the same time.**

This technique is used in **LevelDB** and **RocksDB**, similar storage engines are used in **Cassandra** and **HBase**. *Lucene*, an indexing engine for full text-search used by **elastic search** and **solr***,* uses a similar method for storing its term dictionary.

It only suffers from **one problem**: if the database crashes, the most recent writes are lost. Solution to this is: We can **keep a separate log on disk** to which every write is immediately appended. That log is not in sorted order, but that doesn't matter, because its only purpose is to restore the **memtable** after a crash.

Sstable Lsm Tree Database Index Database Indexing

Edit profile

Written by Suresh Podeti

1.2K Followers

Aspiring Software Architect | SDE III (Golang) @JungleeGames| Ex-Byju's | Co-founder @Rupant Tech. | Ex-Synopsys | I.I.T Kharagpur | 7+ Years of Experience

Recommended from Medium

Augustine Umeagudosi

Database Engineering Part 10: Database Normalization Forms And Their Impact On Redundancy And…

Consider a retail database that stores information about customers, products, and orders. Without normalization, a single table might be…

13 min read · Jan 28, 2024

5